Programming Project # 1 Magician’s Guild Using Linked Lists

Work in groups of 3-4 for this project. Submit one set of deliverables per group.

Your neighbor is an agent for a group of magicians. People call her to book magicians for holidays. She would like to use her new computer to keep track of the jobs she schedules for the magicians she manages, so she hires you to write the program.

# Input

1. The list of magician names is in text file such as "MagicianNames.txt". The names should be listed one per line, and have a maximum of 20 characters. Read the names from the files and create an unsorted linked list from them. You determine how many names to start with (you will provide a method for adding new names later).
2. The list of holidays is in text file such as "Holidays.dat". Again, the names are in listed one per line and have a maximum of 20 characters. Read the holidays from the files and create an unsorted linked list from them. You determine how many names to start with (you will provide a method for adding new names later).
3. The current schedule (retained data from the previous executions of the program) is in file such as "Bookings.dat". You determine the format of this file as part of your assignment. Read the bookings from the files and create an unsorted linked list from them. You will start with no bookings (first time) and add new bookings to the file and the linked list.
4. The user (your neighbor the agent) inputs commands from the keyboard, in response from program prompts, as described under Command Processing below.

# Output

1. Prompts, menus, and responses to user commands are to be written to the screen, as described in the Processing instructions below.
2. A summary of each command's transaction must be written to a text file such as "Magic.trn". You determine the format of the information in this file; it should be labeled and formatted clearly.
3. File "Bookings” must be rewritten to contain the updated magician schedule information as you book events.

# Command Processing (using the lists)

The program must process the commands described below. You may determine the details of the user interface; it must be relatively "friendly" and usable.

## BOOKINGS

Prompt the user for the name of the customer who wants to schedule a magician and for the name of the holiday to be scheduled. Check to see if there is a magician free for this holiday. You should sequence through the magicians in the order in which you read them in. If a magician is available, book the magician (add to bookings list and file); then print out the name of the magician, the holiday, and the name of the customer. If a magician is not available, print out a message indicating that the customer and holiday have been put on a waiting list.

## CANCEL

Prompt the user for the customer name and holiday. Delete the booking of a magician for the listed holiday. Update the booking of the magician who was going to perform for the occasion. NOTE: I recommend rebuilding the bookings file whenever you cancel a booking. Open the file for output and rebuild it from the bookings linked list.

*SIGNUP (new magician)*

Prompt the user for the name of the new magician who is signing up with the agent. Append to the file containing the names and add to the Magician linked list.

## STATUS

Prompt the user for the name of either a magician or holiday. Print out the appropriate schedule, appropriately formatted and labeled.

REPLACE

Prompt for enough information to find the booking in question and replace the current Magician with the next available Magician from the list. NOTE: I recommend rebuilding the bookings file whenever you replace a magician. Open the file for output and rebuild it from the bookings linked list.

*QUIT*

Save the updated data to the files, and then terminate the program.

# Data Structures

You need data structures for storing each of the following:

1. A linked list of bookings. To simply the process of adding, just add new bookings to the end of the list (and the file via append).
2. A linked list node for each magician. (You may assume that there are at most ten magicians.) Each list is the profile of one magician. Each list element contains the name of the magician, the name of the customer who made the booking and the holiday. To simplify the process of adding, just add each new magician to the end of the list (and the file via append). If you have time and the inclination, create the list as a sorted linked list by Magician name.
3. You may use arrays within your program for certain other things (like the holiday list), but not for Magicians or Bookings.
4. You will need the following file data structures
   1. A file to contain the names of the magicians
   2. A file containing the booking information (so you can reload the bookings with each run of your system).
   3. A transaction file for each booking, each magician addition, any booking change (delete, replace).

# Testing

This project can be complicated. You should use both top-down and bottom-up testing. You need to execute the program more than one time. Make a hard copy of the output file ("Magic.trn") after each test run, as the program rewrites this file each time it is executed.

# Deliverables

* Listing of the program's source code.
* Hard copies of file "Magic.trn" from some of the final test executions.
* Implemented test plan and test drivers if appropriate.

NOTES: I will show you an example of how I implemented this project. This is only an example. Feel free to improvise and be creative with your design. You may add other features or even modify your approach to any of the requirements, as long as you demonstrate command of linked lists and their operations, along with file I/O processing.